P1_Wine

Huseyin Coskun/Ruben Garzon

15 Nov 2014

We will first read the Dataset obtained from UCI ML repository https://archive.ics.uci.edu/ml/datasets/Wine

```
wine <- read.csv("../Datasets/Wine/wine.data", header=FALSE)
colnames(wine) <- c("class","Alcohol","Malic Acid","Ash","Alcalinity of Ash","Magnesium","Total Phenols
```

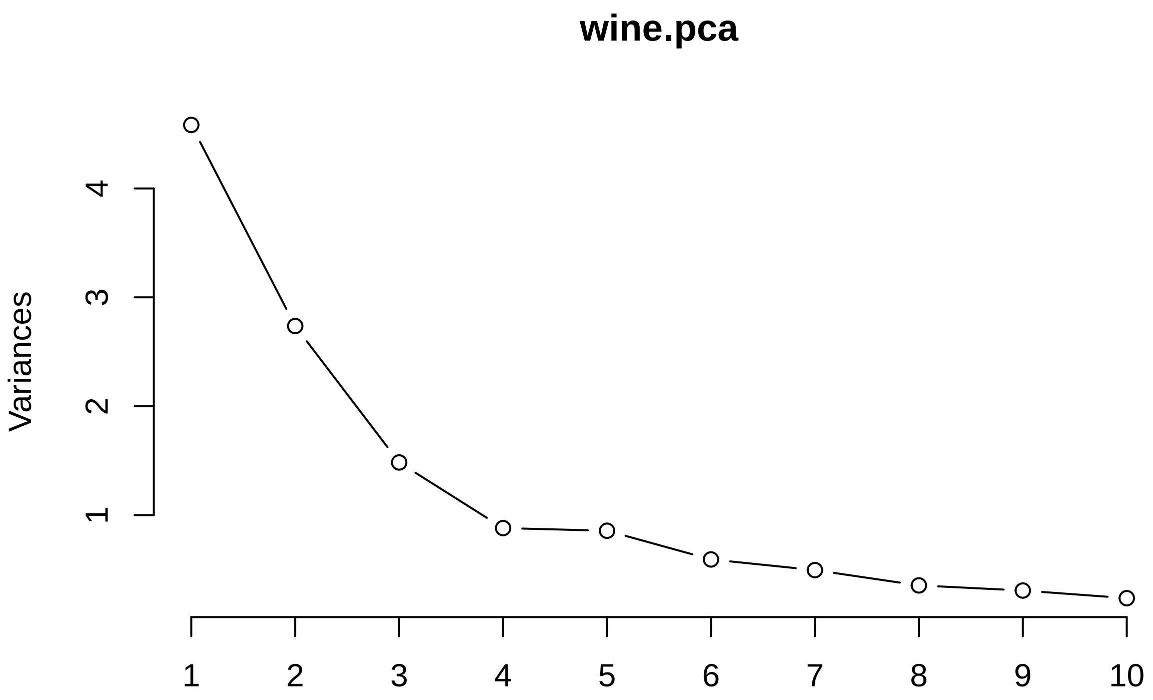
We can plot the data with PCA to explore the distribution of classes (UNIFINISHED)

```
# log transform
log.wine <- log(wine[, 2:14])


# apply PCA - scale. = TRUE is highly
# advisable, but default is FALSE.
wine.pca <- prcomp(log.wine,
                 center = TRUE,
                 scale. = TRUE)
summary(wine.pca)
```

```
## Importance of components:
##                          PC1   PC2   PC3    PC4    PC5    PC6    PC7
## Standard deviation     2.141 1.654 1.218 0.9386 0.9254 0.7699 0.7036
## Proportion of Variance 0.353 0.210 0.114 0.0678 0.0659 0.0456 0.0381
## Cumulative Proportion  0.353 0.563 0.677 0.7449 0.8108 0.8564 0.8945
##                           PC8    PC9   PC10   PC11   PC12    PC13
## Standard deviation     0.5953 0.5544 0.4872 0.4500 0.3886 0.34524
## Proportion of Variance 0.0273 0.0236 0.0183 0.0156 0.0116 0.00917
## Cumulative Proportion  0.9217 0.9454 0.9636 0.9792 0.9908 1.00000
```

```
plot(wine.pca, type = "l")
```



We will first create the training and test with alpha = 0.6 (same value used in the paper)

```r
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
inTrain <- createDataPartition(wine$class, p=0.6, list=FALSE)
trainingSet <- wine[inTrain,]
testSet <- wine[-inTrain,]
```

We would need to scale the variables, but svm seems to do this for us, so at the moment we will not do this.

We apply SVM for prototype selection. We will choose the support vectors as prototypes, although some improvement could be done.

```r
library(e1071)
library(caret)
svmfit=svm(class~., data=trainingSet, kernel="linear", cost=10,scale=FALSE)
prototypes<-trainingSet[svmfit$index,]
```

We now compute the Disimilarity matrix using Euclidean distance The method dist is returning a vector with the lower triangle of the computed distances. This is not easy because we need to concatenate samples matrix + prototype matrix and then select only the elements of the resulting vector from applying dist that we are interested in.

```r
distances <- dist(rbind(trainingSet[,-1],prototypes[,-1]),method="euclidean")
elements <- nrow(trainingSet)*nrow(prototypes)
lowindex <- nrow(trainingSet)-1
upindex <- lowindex + elements -1
relevantDistances <- distances[lowindex:upindex]
dissimilarities <- matrix(relevantDistances,nrow=nrow(trainingSet),ncol=nrow(prototypes))
```

Now we should apply QDA to the dissimilarity training space
Question, in the paper they compare by number of prototypes, should we also do all this tests with different number of prototypes?